PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2000-057329
(43)Date of publication of application : 25.02.2000

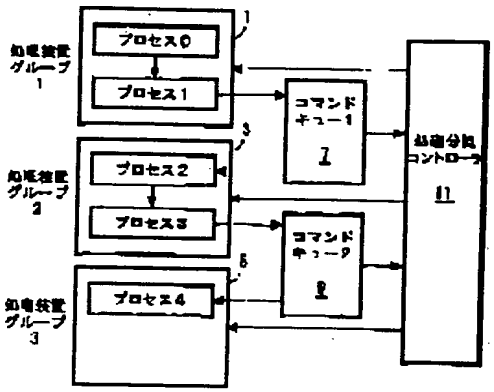
(51)Int.Cl. G06T 1/20
G06F 15/16

(21)Application number : 10-218852
(22)Date of filing : 03.08.1998

(71)Applicant : INTERNATL BUSINESS MACH CORP <IBM>
(72)Inventor : KAWASE KATSURA
MORIYAMA TAKAO
NAKAMURA HIDEFUMI

(54) METHOD FOR DYNAMICALLY CHANGING PROCESSING ALLOTMENT AND COMPUTER

(57)Abstract:
PROBLEM TO BE SOLVED: To draw out the performance of a whole system as much as possible by detecting the increase/decrease of a feature variable in a quene for transmitting/receiving processed results between plural groups and changing the allotment of processing in the respective groups based on the increase/ decrease of the feature variable.
SOLUTION: A processing allotment controller 11 monitors command queues 1 (7), 2 (9) and detects the increase/decrease of a feature variable to be an index in the case of changing the processing allotment. When the feature variable of the command quene 1 (7) is reduced to more than a prescribed threshold, the controller 11 instructs a processor group 1 (1) to execute only a process 0 and instructs a processor group 2 (3) also to execute a process 1. When the feature variable of the command queue 2 (9) is reduced to more than the prescribed threshold, the allotment processes are adjusted between both processor groups 2 (3) and 3 (5).



LEGAL STATUS

[Date of request for examination]	27.12.1999
[Date of sending the examiner's decision of rejection]	27.02.2001
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]	
[Date of final disposal for application]	
[Patent number]	3224782
[Date of registration]	24.08.2001
[Number of appeal against examiner's decision of rejection]	
[Date of requesting appeal against examiner's decision of rejection]	
[Date of extinction of right]	

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開2000－57329
（P2000－57329A）
(43)公開日　平成12年２月25日(2000.2.25)

(51)Int.Cl.'	識別記号	FI		テーマコード'(参考)
G06T 1/20		G06F 15/66	L	5B045
G06F 15/16	370	15/16	370N	5B057

審査請求　未請求　請求項の数12　OL　（全 10 頁）

(21)出願番号　特願平10－218852

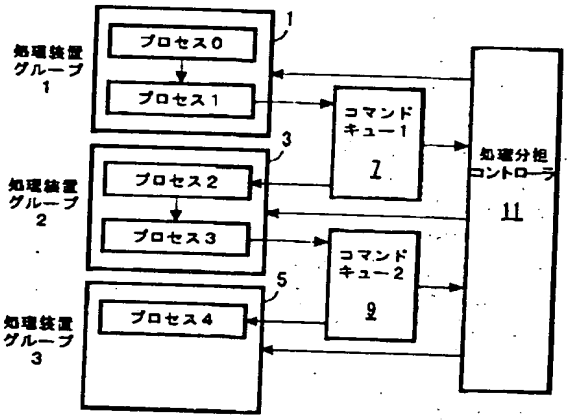
(22)出願日　平成10年８月３日(1998.8.3)

(71)出願人　390009531
インターナショナル・ビジネス・マシーンズ・コーポレイション
INTERNATIONAL BUSINESS MASCHINES CORPORATION
アメリカ合衆国10504、ニューヨーク州　アーモンク　（番地なし）
(74)代理人　100086243
弁理士　坂口　博　（外１名）

最終頁に続く

(54)【発明の名称】　処理分担動的変更方法及びコンピュータ

(57)【要約】
【課題】処理量及び処理能力が予測不可能なマルチ・プロセッシング・システムにおいて、処理分担を動的に変更する。
【解決手段】少なくとも２つのグループに分けることができる複数の処理装置を有するコンピュータにおいて、各グループにおける、第１段から第ｎ段までの一連の処理の分担を動的に変更する本発明は、グループ間の処理結果の受渡しのためのキューにおける特徴量の増減を検出するステップと、特徴量の増減に基づき、各グループにおける処理の分担を変更する変更ステップとを含む。キューに格納されたデータの特徴量は、処理量に関係する値を表すものであり、この特徴量を参照しつつ処理分担を変更すれば、キューが満杯又は空になることはほとんど生じなくなる。



【特許請求の範囲】
【請求項１】少なくとも２つのグループに分けることができる複数の処理装置を有するコンピュータにおいて、各前記グループにおける、第１段から第ｎ段までの一連の処理の分担を動的に変更する方法であって、
前記グループ間の処理結果の受渡しのためのキューにおける特徴量の増減を検出するステップと、
前記特徴量の増減に基づき、各前記グループにおける処理の分担を変更する変更ステップと、
を含む処理分担動的変更方法。
【請求項２】前記特徴量は前記キューに格納された情報の量である、請求項１記載の処理分担動的変更方法。
【請求項３】前記特徴量は、グラフィックスに関連する処理の場合、前記キューに格納された頂点データの数である、請求項１記載の処理分担動的変更方法。
【請求項４】前記変更ステップは、
前記特徴量が所定のしきい値以上増加した場合には、前記グループ間の処理分担の境界である第ｉ段（１≦ｉ＜ｎ）までの処理を実施していたグループの担当を第ｉ段より後段の処理まで当該グループの担当と設定するステップ、
を含む請求項１記載の処理分担動的変更方法。
【請求項５】前記変更ステップは、
前記特徴量が所定のしきい値以上減少した場合には、前記グループ間の処理分担の境界である第ｉ段（１＜ｉ≦ｎ）までの処理を実施していたグループの担当を第ｉ段より前段の処理までを当該グループの担当と設定するステップ、
を含む請求項１記載の処理分担動的変更方法。
【請求項６】前記処理結果は、何段目までの処理を実施したかについての情報を含む、請求項１記載の処理分担動的変更方法。
【請求項７】前記キューの使用量が上限に達していないか検査するステップと、
前記キューの使用量が上限に達している場合、前記グループ間の処理分担の境界である第ｉ段（１≦ｉ＜ｎ）までの処理を実施しているグループに属する処理装置が前記キューの最後尾の処理結果を取り出し、前記第ｉ段より後段の処理まで実施した後に、前記キューに当該処理結果を格納するステップと、
をさらに含む請求項１記載の処理分担動的変更方法。
【請求項８】少なくとも２つのグループに分けることができ、各前記グループにおける、第１段から第ｎ段までの一連の処理の分担が設定される、複数の処理装置と、
前記グループ間の処理結果の受渡しのためのキューと、
前記キューにおける特徴量の増減を検出し、前記特徴量の増減に基づき、各前記グループにおける処理の分担を変更するコントローラと、
を有するコンピュータ。
【請求項９】前記コントローラは、
前記特徴量が所定のしきい値以上増加した場合には、前記グループ間の処理分担の境界である第ｉ段（１≦ｉ＜ｎ）までの処理を実施していたグループの担当を第ｉ段より後段の処理まで当該グループの担当と設定する、
請求項８記載のコンピュータ。
【請求項１０】前記コントローラは、
前記特徴量が所定のしきい値以上減少した場合には、前記グループ間の処理分担の境界である第ｉ段（１＜ｉ≦ｎ）までの処理を実施していたグループの担当を第ｉ段より前段の処理までを当該グループの担当と設定する、
請求項８記載のコンピュータ。
【請求項１１】前記コントローラは、
前記キューの使用量が上限に達していないか検査し、
前記キューの使用量が上限に達している場合、前記グループ間の処理分担の境界である第ｉ段（１≦ｉ＜ｎ）までの処理を実施しているグループに属する処理装置に、前記キューの最後尾の処理結果を取り出し、前記第ｉ段より後段の処理まで実施した後に、前記キューに当該処理結果を格納するように命ずる、
請求項８記載のコンピュータ。
【請求項１２】少なくとも２つのグループに分けることができる複数の処理装置を有するコンピュータに、各前記グループにおける、第１段から第ｎ段までの一連の処理の分担を動的に変更させるプログラムを格納した記憶装置であって、
前記プログラムは、前記コンピュータに、
前記グループ間の処理結果の受渡しのためのキューにおける特徴量の増減を検出するステップと、
前記特徴量の増減に基づき、各前記グループにおける処理の分担を変更する変更ステップと、
を実行させる、記憶媒体。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は、マルチプロセッサ・システムにおける処理分担の動的変更に関し、より詳しくは、コンピュータにおけるホスト・プロセッサとグラフィックス・アダプタにおける処理分担の動的変更に関する。
【０００２】
【従来の技術】ＯｐｅｎＧＬやＤｉｒｅｃｔ３Ｄといったポリゴンベースの三次元グラフィックスの場合、全体のパフォーマンスを決める主な要因は以下のようなものがある。
（１）ＡＰＩ
アプリケーションからＡＰＩを介して描画コマンドを発行する速度
（２）ジオメトリ（Geometry）処理
三角形分割／座標変換／照度計算といったジオメトリ処理の速度
（３）セットアップ（Setup）処理

三角形の辺／面に沿った色／Ｚ座標値／テクスチャ座標値の勾配計算の速度
（４）ラスタ処理
三角形内部のピクセルの色／Ｚ座標値／テクスチャ座標値を補間して求め、それらをフレーム・バッファへ読み書きする速度
【０００３】（１）は、最も処理量が多い方法であるところの、頂点ごとにＡＰＩを呼び出す方法を用いても、一頂点当たり数十クロックしかかからないので、（１）が問題となることはない。
【０００４】（４）は一秒間に何ピクセル描画できるか（ピクセル・レート（pixel rate）と呼ぶ）に相当する。このピクセル・レートはポリゴン・レート（polygon rate、後述）とは無関係であり、画面サイズ（例えば６４０×４８０や１０２４×７６８といった一画面が何ピクセルから構成されるか。）とフレーム・レート（frame rate。一秒間に何コマ表示を行うか。ＣＲＴのリフレッシュ・レート（Refresh rate）とは異なる。一般に、１２－６０フレーム／秒程度必要。）と画面上での平均的な重なり（通常３回程度）によって要求量が決定されるものである。最近のグラフィックス・アダプタではＳＸＧＡ（１２８０×１０２４ピクセル）程度の画面サイズまで、ほぼ問題ない程度になりつつある。
【０００５】（２）及び（３）のジオメトリとセットアップ処理のパフォーマンスは直接一秒間に何ポリゴン処理できるか（先に述べたポリゴン・レート）に相当する。セットアップ処理は、ジオメトリ処理の一部とされることも多いので、ここではジオメトリ処理として扱うことにする。ジオメトリ処理は数多くの浮動小数点演算が必要である。これには１頂点あたりの処理に数百から数千クロックかかる。従って、ホスト・プロセッサの処理能力だけでは不十分な場合が多い。例えば、１０Ｍ個の頂点を１秒間に処理する場合、１頂点処理するのに１０００クロック必要ならば、１０Ｇクロック／秒のプロセッサが必要になってしまう。よって、グラフィックス・アダプタ上にジオメトリ処理専用の演算器を設けることが多い。また処理の条件、例えば光源の数や光源の種類等によって処理量が大きく変化する。
【０００６】ところで、ホスト・プロセッサは描画コマンド（Graphics Command）の列を主記憶装置に蓄える。この描画コマンドの列をコマンド・キュー（Command Queue）という。グラフィックス・アダプタはＤＭＡを用いてコマンド・キューの内容を内部に取り込み、処理を施して表示装置に表示する。このコマンド・キューはＤＭＡ転送を行う必要上物理的に主記憶装置内に又はグラフィックス・アダプタ上に存在しなければならない。従って、コマンド・キューの大きさには制限がある。このコマンド・キューが処理の途中で満杯になったり、空になったりすると、ホスト・プロセッサ又はグラフィックス・アダプタが停止するので、全体としてのパフォーマンスが落ちてしまう。コマンド・キューが満杯（Ｆｕｌｌ）の場合には、ホスト・プロセッサはこれ以上コマンド・キューに書き込むことができないので、それに空きが生じるまでホスト・プロセッサは処理を進めることができない。また、コマンド・キューが空（Ｅｍｐｔｙ）である場合には、グラフィックス・アダプタは処理を行うことができない。
【０００７】ホスト・プロセッサの処理速度とグラフィックス・アダプタの処理速度が等しければコマンド・キューは満杯又は空になることはないが、以下の理由で両者の処理速度を均等にすることはできなかった。
（ａ）表示のために使用できるホスト・プロセッサの処理能力が予測困難である。ホスト・プロセッサの種類／動作周波数／個数が多様である。また、表示以外で使用するホスト・プロセッサの負荷が予測困難あり、動的に変化する。
（ｂ）上述のジオメトリ処理のように、描画コマンドのホスト・プロセッサ上での処理量（ワークロード：work load）は、現在の状態やデータに依存して動的に変化するので、予測困難である。例えばクリッピング（clipping）によって頂点の数は増減する。
（ｃ）描画コマンドのグラフィックス・アダプタ上での処理量は、現在の状態やデータに依存して動的に変化するので、予測困難である。
【０００８】ホスト・プロセッサの処理能力と処理量をそれぞれP_h，L_hとし、グラフィックス・アダプタの処理能力と処理量をそれぞれP_a，L_aとすると、$L_h/P_h=L_a/P_a$が成り立てば、コマンド・キューが空になったり満杯になったりすることなく処理が進められるが、L_h、P_h、L_a及びP_aはいずれも予測不可能であり、必ずしもシステムの性能を最大限引き出すことができていなかった。
【０００９】特開平２－２７５５８１号公報は、機能毎にその機能を使用する際に必要な時間が予め分かっている場合に、利用者が使用する幾つかの機能をオン／オフするごとに、複数のプロセッサの処理分担を変更して、システム全体の処理速度を向上させる技術を開示している。しかし、機能を使用する際に必要な時間は、処理するデータに依存するため、この公報の前提では処理分担を適切に変更できない。また、ホスト・プロセッサはマルチタスクＯＳの環境にあることが多く、グラフィックスに割り当てられる計算能力は時々刻々と変化するため、この点においてもこの公報の前提は適切ではない。さらに、この公報では、機能のオン／オフの全ての組み合わせに対応する処理分担表を作っておく必要があるが、実際の環境においてはオン／オフできる機能の数は膨大になり実際的ではない。
【００１０】
【発明が解決しようとする課題】よって、本発明では、L_h、P_h、L_a及びP_aがいずれも予測不可能である環境

において、$L_h/P_h=L_a/P_a$に近づけることができるようなコンピュータ・システムを提供することである。

【００１１】また、$L_h/P_h=L_a/P_a$に近づけることにより、システム全体のパフォーマンスをできる限り引き出すことができるようにすることも目的である。

【００１２】さらに、将来のホスト・プロセッサの処理能力の向上に適応できるようにし、製品寿命を伸ばすことも目的である。

【００１３】さらに、コマンド・キューが満杯になるような事態が生じても、ホスト・プロセッサの処理を停止せずにシステム全体のパフォーマンスを下げないようにすることも目的である。

【００１４】

【課題を解決するための手段】少なくとも２つのグループに分けることができる複数の処理装置を有するコンピュータにおいて、各グループにおける、第１段から第ｎ段までの一連の処理の分担を動的に変更する本発明は、グループ間で処理結果を受渡すためのキューにおける特徴量の増減を検出するステップと、特徴量の増減に基づき、各グループにおける処理の分担を変更する変更ステップとを含む。キューに格納されたデータの特徴量は、処理量に関係する値を表すものであり、この特徴量を参照しつつ処理分担を変更すれば、キューが満杯又は空になることはほとんど生じなくなる。

【００１５】例えば、この特徴量は、キューに格納された情報の量又はキューのサイズ（長さ）であっても、グラフィックスに関連する処理の場合、キューに格納された頂点データの数であってもよい。

【００１６】また、先に述べた変更ステップは、特徴量が所定のしきい値以上増加した場合には、グループ間の処理分担の境界である第ｉ段（$1\leq i<n$）までの処理を実施していたグループの担当を第ｉ段より後段の処理まで当該グループの担当と設定するステップを含むようにすることも考えられる。後段の処理とは、第ｉ＋１段以降の処理である。また、特徴量が所定のしきい値以上減少した場合には、グループ間の処理分担の境界である第ｉ段（$1<i\leq n$）までの処理を実施していたグループの担当を第ｉ段より前段の処理までを当該グループの担当と設定するステップを実行するようにすることも考えられる。また、前段の処理とは、第ｉ－１段以前の処理である。

【００１７】またこのように処理分担を動的に変えると、後段の処理を実施するグループは、その段階から処理を実施すべきかという情報が必要になる場合がある。よって、処理結果は、何段目までの処理を実施したかについての情報を含むようにすることも考えられる。

【００１８】また、キューの使用量が上限に達していないか検査するステップと、キューの使用量が上限に達している場合、グループ間の処理分担の境界である第ｉ段（$1\leq i<n$）までの処理を実施しているグループに属する処理装置がキューの最後尾の処理結果を取り出し、第ｉ段より後段の処理まで実施した後に、キューに当該処理結果を格納するステップをさらに含むようにすることも考えられる。これにより、コマンド・キューが満杯になるような事態が生じても、ホスト・プロセッサの処理を停止せずにシステム全体のパフォーマンスを下げないようにすることができる。

【００１９】本発明を実施したコンピュータは、少なくとも２つのグループに分けることができ、各前記グループにおける、第１段から第ｎ段までの一連の処理の分担が設定される、複数の処理装置と、グループ間の処理結果の受渡しのためのキューと、キューにおける特徴量の増減を検出し、特徴量の増減に基づき、各グループにおける処理の分担を変更するコントローラとを有する。

【００２０】以上本発明の構成を説明したが、各ステップを実行するようなプログラムにて本発明を実施することも可能である。その際、プログラムはＣＤ－ＲＯＭやフロッピー・ディスク等の記憶媒体に記憶されたり、ハードディスクやＲＯＭ等の記憶装置又はデバイスに記憶される場合がある。また、本発明の処理を実施するような専用の回路又は装置を実施することも可能である。

【００２１】

【発明の実施の形態】図１にパイプライン式に処理すべきプロセス群を示す。図１では説明を簡単にするため、プロセス０乃至４を示しているが、これより多いプロセスが存在する場合及び少ないプロセスが存在する場合の両方が考えられる。グラフィックスに関する処理もこのようなパイプライン式に実施する必要がある。

【００２２】図１のようなプロセス０乃至４を３つの処理装置グループで処理する場合の一例を図２に示している。処理装置グループ１（１）はプロセス０及び１を担当しており、処理装置グループ２（３）はプロセス２及び３を担当しており、処理装置グループ３（５）はプロセス４を担当している。処理装置グループとしているのは、各処理装置グループに含まれるプロセッサの数は１つでも複数でもよいからである。この処理装置グループ間のデータの交換にはコマンド・キュー１（７）及びコマンド・キュー２（９）が用いられる。

【００２３】コマンド・キュー１（７）は処理装置グループ１及び２の間のデータの交換に、コマンド・キュー２（９）は処理装置グループ２及び３の間のデータの交換に用いられる。このコマンド・キューは、処理装置グループが用いるメモリの一部に設けられるようにしても、別個設けるようにしてもよい。コマンド・キューは、コマンド・キューを挟んだ両側の処理装置グループの処理速度が多少変動しても、ある程度の時間空になったり満杯になったりしないような大きさが必要である。これは、処理装置グループの他のプロセスやスレッドにより処理能力が変動したり、両側の処理装置グループの処理量は離散的にしか分割できないため、正確にはL_1

／P_1＝L_2／P_2（１及び２は処理装置グループ１及び２を示す）を保つことはできないからである。

【００２４】また、処理分担コントローラ１１は、コマンド・キュー１及び２を監視して、処理分担を変更する際の指標となる特徴量の増減を検出する。処理分担コントローラ１１は必要に応じて処理分担を変更するよう処理装置グループに命ずる。なお、図２では１つの処理分担コントローラ１１を設けているが、各処理装置グループ内で同様の機能を実施するようにすることも可能である。

【００２５】処理装置グループ１（１）は、プロセス０及びプロセス１を実行し、コマンド・キュー１（７）にプロセス１の処理結果を順次書き込み、例えばＤＭＡ（図示せず）のような機構を介してプロセス２を実施する処理装置グループ２（３）にデータを渡す。処理装置グループ２（３）は、コマンド・キュー１（７）における処理結果を順次用いてプロセス２及びプロセス３を実施し、コマンド・キュー２（７）にプロセス３の処理結果を書き込み、同じようにしてプロセス４を実施する処理装置グループ３（５）にデータを渡す。処理装置グループ３はコマンド・キュー２（９）における処理結果を順次用いてプロセス４を実施する。

【００２６】図３に、コマンド・キュー１（７）における特徴量が所定のしきい値以上減少した場合を示す。コマンド・キュー１（７）における特徴量が所定のしきい値以上減少した場合には、処理装置グループ１（１）の負荷が大きくなったか又は処理能力が落ちたか、若しくは処理装置グループ２（３）の負荷が軽くなったか又は処理能力が上がったか、である。とにかく、このまま放置するとコマンド・キュー１（７）が空になって、処理装置グループ２（３）が遊んでしまう。そこで、処理分担コントローラ１１は、処理装置グループ１にプロセス０のみ実行するように命じ、処理装置グループ２（３）にプロセス１も実行するように命ずる。図３はこのような処理分担の変更を実施した後の状態を示している。もし、この後コマンド・キュー２（９）における特徴量が所定のしきい値以上減少するような場合には、再度処理装置グループ２（３）及び処理装置グループ３（５）の間で担当プロセスの調整がなされる。

【００２７】通常のコンピュータにおけるグラフィックス処理を考えた場合には、図２及び図３における処理装置グループは２つで、ホスト・プロセッサ（１又は複数）のグループとグラフィックス・アダプタのグループに分けられる。図４に本発明のコンピュータの例を示す。ホスト・プロセッサ２１はメモリ・コントローラ２３に接続している。メモリ・コントローラ２３はメイン・メモリ２５及びバス２７に接続している。バス２７にはグラフィックス・アダプタ２９が接続しており、このグラフィックス・アダプタ２９は表示装置３１に接続している。ホスト・プロセッサ２１は上でも述べているように複数のプロセッサであってもよい。メモリ・コントローラ２３にはＤＭＡコントローラを含む。また、メイン・メモリ２５内には、コマンド・キュー２５ａと、ソフトウエアとして実施されている処理分担コントローラ２５ｃと、本発明ではホスト・プロセッサ２１が処理した頂点データの数（コマンド・キュー２５ａへの入力となる）をカウントするカウンタ１（２５ｂ）とを含む。また、メイン・メモリ２５には実行中の他のソフトウエア（割り当てられたプロセスに関するプログラムを含む）も含まれ、それらは必要に応じて図示しないハードディスク等からロードされる。グラフィックス・アダプタ２９には、図示しないジオメトリ・プロセッサ及びラスタ・プロセッサと、グラフィックス・アダプタ２９が処理した頂点データの数をカウントするカウンタ２９ａと、割り当てられる可能性のある処理プロセスに必要なプログラム（図示せず）を含む。

【００２８】ホスト・プロセッサ２１は、図示しないプログラムを用いて割り当てられたプロセスの処理を実施し、その結果をコマンド・キュー２５ａにメモリ・コントローラ２３を介して書き込む。頂点データを１つ処理するごとにカウンタ１（２５ｂ）を１インクメントする。また、メモリ・コントローラ２３は所定のサイズごとにコマンド・キュー２５ａのデータをバス２７を介してグラフィックス・アダプタ２９に渡す。グラフィックス・アダプタ２９は渡されたデータ（ホスト・プロセッサ２１の処理結果）を用いて割り当てられたプロセスを実施し、処理結果として表示装置３１にグラフィックスを表示する。なお、グラフィックス・アダプタ２９において頂点データが１つ処理されるごとにカウンタ２（２９ａ）を１インクリメントする。

【００２９】ホスト・プロセッサ２１側に設けられた処理分担コントローラ２５ｃは、一定期間ごとにカウンタ１（２５ｂ）及びカウンタ２（２９ａ）の値を取得し、各カウンタ値の増分を用いて後に説明する処理分担の変更を行う。本発明では、ホスト・プロセッサ２１側に処理分担コントローラ２５ｃを設けているが、これはグラフィックス・アダプタ２９側に設けることも可能である。

【００３０】コマンド・キュー２５ａの一例を図５に示す。コマンド・キュー２５ａは、ＤＭＡ時のページ・マッピングの問題や、ホスト・プロセッサ２１が複数のプロセッサを含むような場合のロックの問題から、ページ境界（page boundary）に沿ったページ・サイズ以下のリンク・リスト（Linked List）構造が好ましい。リング・バッファはポインタを格納しており、各ポインタの先には各キュー・エレメントが配置されている。キュー・エレメントは描画コマンド（Ｃｏｍｍａｎｄ）及びデータ（Ｄａｔａ）を含んでおり、ページ・サイズ以下で例えば２－４Ｋバイトである。書き込みポインタ（Ｗｒｉｔｅ　ＰＴＲ）はホスト・プロセッサ２１が書き込ん

だキュー内の最後のキュー・エレメントを指す。読み出しポインタ（Ｒｅａｄ　ＰＴＲ）はグラフィックス・アダプタ２９が次に読み出すべきキュー・エレメントのアドレスを指す。ＤＭＡでキュー・エレメントを読み出すと、読み出しポインタを１つ進める。

【００３１】ホスト・プロセッサ２１とグラフィックス・アダプタ２９の間で処理プロセス間のデータの受渡し方法を予め定義しておき、ホスト・プロセッサ２１がコマンド・キュー２５ａに処理結果として描画コマンドとデータを渡す際に、どの処理プロセスまで実施したかを表すタグを付加しておく。タグには、例えばプロセス２まで実施したということを含めてもよいし、プロセス３から実施しろということを含めても良い。このタグは例えばキュー・エレメント内の描画コマンド内に含める。このようにすれば、グラフィックス・アダプタ２９がどの処理プロセスから残りの処理プロセスを実施すればよいかが分かるので、全体として正しい結果が得られる。

【００３２】処理分担コントローラ２５ｃの処理を説明する前に、処理分担コントローラ２５ｃが処理分担を変更するために参照する特徴量について説明しておく。図４に示したように、本実施例では頂点データの数を特徴量としている。これは、コマンド・キュー２５ａには頂点データ以外にもビットマップ等の二次元イメージを描画する命令も含まれる。この場合にはジオメトリ処理は必要なく素通りし、ラスタ処理部で処理される。ビットマップ等がコマンド・キュー２５ａに入っている場合には、コマンド・キューの大きさは大きくなるし、頂点データに比して処理負荷は軽くなるので、長さ又は大きさを基準にすると、グラフィックス・アダプタ２９への処理分担は不正確になるおそれがある。よって、本実施例ではコマンド・キュー２５ａに含まれる頂点データの数を特徴量として取り扱う。但し、他の状況においてはコマンド・キューの長さや大きさを特徴量とすることが可能な場合も生じえる。

【００３３】特徴量である頂点データの数を把握するために、コマンド・キュー２５ａを走査して、カウントすることも可能である。しかし、本実施例では、ホスト・プロセッサ２１とグラフィックス・アダプタ２９がそれぞれ自分で処理した頂点データを数を、カウンタ１（２５ｂ）及びカウンタ２（２９ａ）でカウントし（それぞれカウント１及びカウント２とする）、一定期間における増分（それぞれ△カウント１及び△カウント２とする）の差によりコマンド・キュー２５ａ内に存在する頂点データの数を把握することにする。

【００３４】例えば、△カウント２が所定のしきい値以上△カウント１より大きい場合には、グラフィックス・アダプタ２９の処理の方が速いので、ホスト・プロセッサ２１に割り当てられている処理プロセスをグラフィックス・アダプタ２９に割り当てないと、コマンド・キュー２５ａが空になってしまうと判断できる。また、△カウント１が所定のしきい値以上△カウント２より大きい場合には、グラフィックス・アダプタ２９の処理の方が遅いので、グラフィックス・アダプタ２９に割り当てられた処理プロセスをホスト・プロセッサ２１に割り当てないと、コマンド・キュー２５ｃが満杯になっていしまうと判断できる。このような判断を、△カウント１及び△カウント２の関数を定義して行うことも出来る。

【００３５】処理分担コントローラ２５ｃの処理フローを図６に示す。まず、特徴量の増減を検出するために、カウンタ１及びカウンタ２の値を取得し、△カウント１及び△カウント２を計算する（ステップ１１０）。そして、△カウント１が△カウント２より第１のしきい値以上大きいか判断する（ステップ１２０）。もし大きい場合には、コマンド・キュー２５ａ内の頂点データは許容できる範囲を超えて増加しているため、満杯になってしまう可能性がある。よって、グラフィックス・アダプタ２９の担当処理プロセスを第ｉ段以降から第ｉ＋ｐ段以降に変更する（ステップ１５０）。ここでｐは正の整数である。なお、グラフィックス・アダプタ２９の担当処理プロセスを減らせば、自動的にホスト・プロセッサ２１の担当分が増加する。

【００３６】一方、ステップ１２０の条件が満たされない場合には、△カウント２が△カウント１より第２のしきい値以上大きいか判断する（ステップ１３０）。もし、この条件が満たされる場合には、コマンド・キュー２５ａ内の頂点データ数が許容範囲を超えて減少しているため、コマンド・キュー２５ａは空になってしまうかもしれない。そこで、ホスト・プロセッサ２１の担当処理プロセスを第ｉ段以前から第ｉ－ｍ段（ｍは正の整数）以前に変更する（ステップ１４０）。

【００３７】そして、所定時間経過後、再度ステップ１１０からの処理を繰り返す。

【００３８】このようにして、特徴量の増減に基づき、各処理装置グループにおける処理の分担を変更する。

【００３９】図６の処理フローは様々な変更が可能である。△カウント１及び△カウント２の差である△カウントの増減によって、処理分担を変更するようにしてもよい。また、第１及び第２のしきい値は同じでも異なっていても良い。また、ｐ及びｍは固定であっても△カウント１及び△カウント２の差の大きさによって変化させてもよい。

【００４０】ステップ１６０における所定時間は、コマンド・キュー２５ａを空にしないような周期、具体的には以下のような条件で決めることができる。すなわち、現在のコマンド・キュー２５ａの内容を全てグラフィックス・アダプタ２９に転送するのに必要な時間をＸとし、一つのキュー・エレメントをホスト側にて最高速度（最大限グラフィックス・アダプタ２９に処理を実行させる場合）で作成する時間をＣとした場合、処理分担を変更する周期である、この所定時間ＴはＴ＜Ｘ－Ｃであ

ればよい。

【００４１】なお、ホスト・プロセッサ２１に処理すべき描画コマンドが残っているのに、コマンド・キュー２５ｃ（サイズは通常数十Ｋバイトから数Ｍバイト）の使用量が上限に達してしまうと、ホスト・プロセッサ２１は処理を続行することができなくなる。よって、例えば図６におけるステップ１２０のようなタイミングで、コマンド・キュー２５ａが満杯かどうかを検査して、もし満杯であれば、処理分担コントローラ２５ｃはホスト・コンピュータ２１に取り戻し命令を出力し、図７のような処理が実施される。タイミングは他の場合であってもよく、ホスト・プロセッサ２１が書き込みの際に自分で判断して図７のような処理を開始しても良い。

【００４２】まず、ホスト・コンピュータ２１は、コマンド・キュー２５ａ内の最後尾の処理結果を取り出す（ステップ２１０）。これは、図８のように、リング・バッファの元の書き込みポインタ（ＷＲＩＴＥ＿ＰＲＴ）から最後尾のキュー・エレメント７を取り戻すものである。なお、書き込みポインタの位置を一つ戻す。本実施例では最後尾の１つのキュー・エレメントを取り出すことにしているが、複数のキュー・エレメントを取り出すことにしてもよい。そして、グラフィックス・アダプタ２９担当の処理プロセスの一部を、取り戻した処理結果に対して実施する（ステップ２３０）。例えば第ｉ段まで担当していた場合には、第ｉ＋１段以降を実施する。ホスト・プロセッサ２１はどの処理プロセスまで実施したかはキュー・エレメントの描画コマンド部分に付されたタグを見れば分かるので、その後の段階の処理プロセスを実施する。図８の例では、後段の処理プロセスを実施した結果が、キュー・エレメント７'として示されている。どの段階まで処理プロセスを実施するかは任意である。但し、場合によっては、再度キュー・エレメント７'を取り戻して処理する場合も生じえる。最後に、実施終了した処理プロセスのタグを付して、再度処理結果をコマンド・キュー２５ａに入力する（ステップ２３０）。このタグは、次にグラフィックス・アダプタ２９が開始すべき処理プロセスに関する情報でもよい。図９にステップ２３０終了後の状態を示す。この図ではまだキュー・エレメント０がグラフィックス・アダプタ２９へ転送されていないので、コマンド・キュー２５ａが満杯ということになってしまうが、キュー・エレメント０の転送が終了していれば、ホスト・プロセッサ２１は次の処理に取りかかることができる。以上のようにして、コマンド・キューが万が一満杯になった場合に対処する。

【００４３】以上本発明の一実施例を説明したが、本発明は上の実施例に限定されない。例えば、コマンド・キューの構造は図５のような構造に限定されない。コマンド・キューはキュー・エレメントをチェーン化して構成することも可能である。また、処理装置グループ内の複数のプロセッサは同じ能力でなくともよい。また、処理分担コントローラはソフトウエアで実施しても専用の回路などを用意しても良い。

【００４４】

【効果】L_1、P_1、L_2及びP_2（それぞれ第１処理装置グループの処理量、処理能力、第２処理装置グループの処理量、処理能力）がいずれも予測不可能である環境において、$L_1/P_1=L_2/P_2$に近づけることができるようなコンピュータ・システムを提供することができた。

【００４５】また、$L_1/P_1=L_2/P_2$に近づけることにより、システム全体のパフォーマンスをできる限り引き出すことができるようにすることもできた。

【００４６】さらに、将来のホスト・プロセッサの処理能力の向上に適応できるようにし、製品寿命を伸ばすこともできた。

【００４７】さらに、コマンド・キューが満杯になるような事態が生じても、コマンド・キューに書き込みを行う処理装置グループの処理を停止せずにシステム全体のパフォーマンスを下げないようにすることもできた。

【図面の簡単な説明】

【図１】パイプライン式の処理プロセスの例である。

【図２】複数の処理装置グループにおいて処理プロセスを分担する際の機能ブロック図である。

【図３】図２の状態から処理分担を変更した場合の図である。

【図４】グラフィックス処理の処理プロセスをホスト・プロセッサとグラフィックス・アダプタとで分担する場合の機能ブロック図である。

【図５】コマンド・キュー構造の例である。

【図６】処理分担コントローラの処理フローの例である。

【図７】コマンド・キューが満杯になった際のホスト・プロセッサの処理フローの例である。

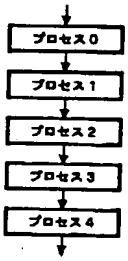
【図８】図７の処理を模式的に示した図である。

【図９】図７の処理結果を模式的に示した図である。

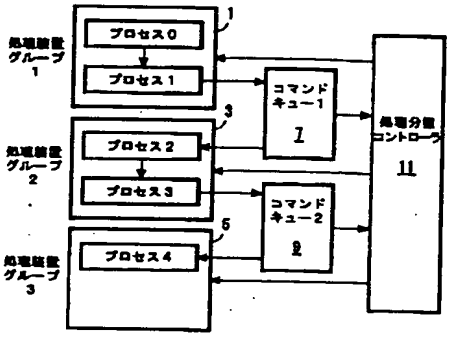
【符号の説明】

１、３、５　処理装置グループ
７、９　コマンド・キュー
１１　処理分担コントローラ
２１　ホスト・プロセッサ
２３　メモリ・コントローラ
２５　メイン・メモリ
２５ａ　コマンド・キュー
２５ｂ　カウンタ１
２５ｃ　処理分担コントローラ
２７　バス
２９　グラフィックス・コントローラ
２９ａ　カウンタ２
３１　表示装置

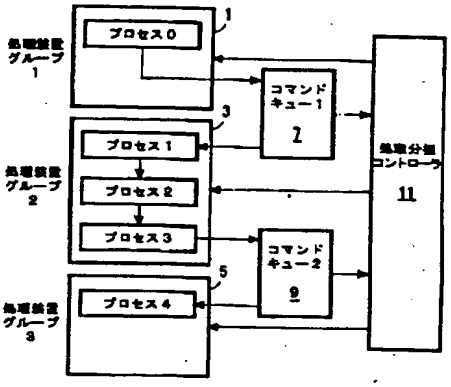
【図1】



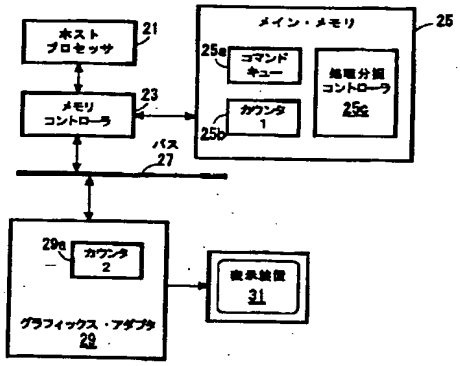
【図2】



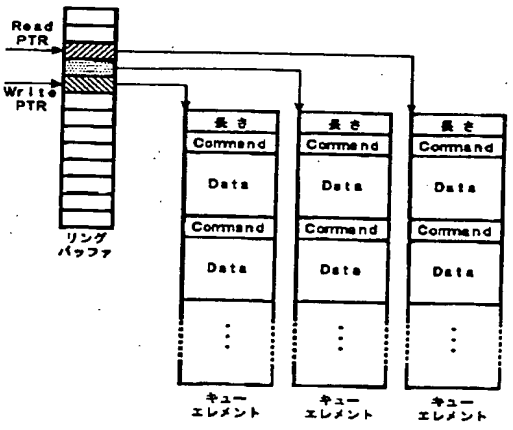
【図3】



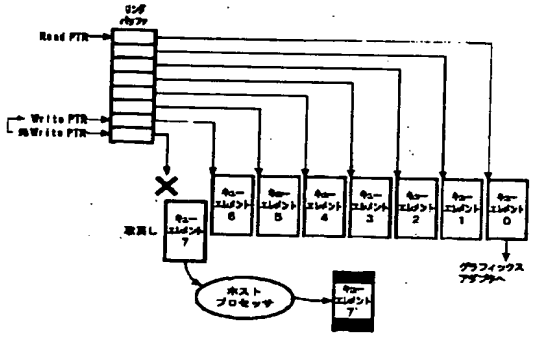
【図4】



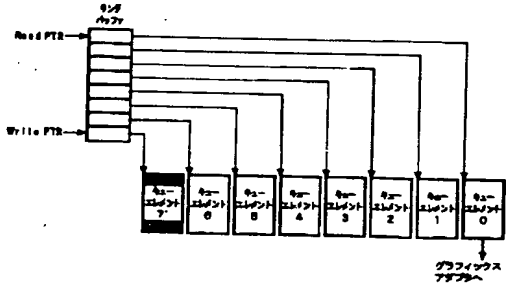
【図5】



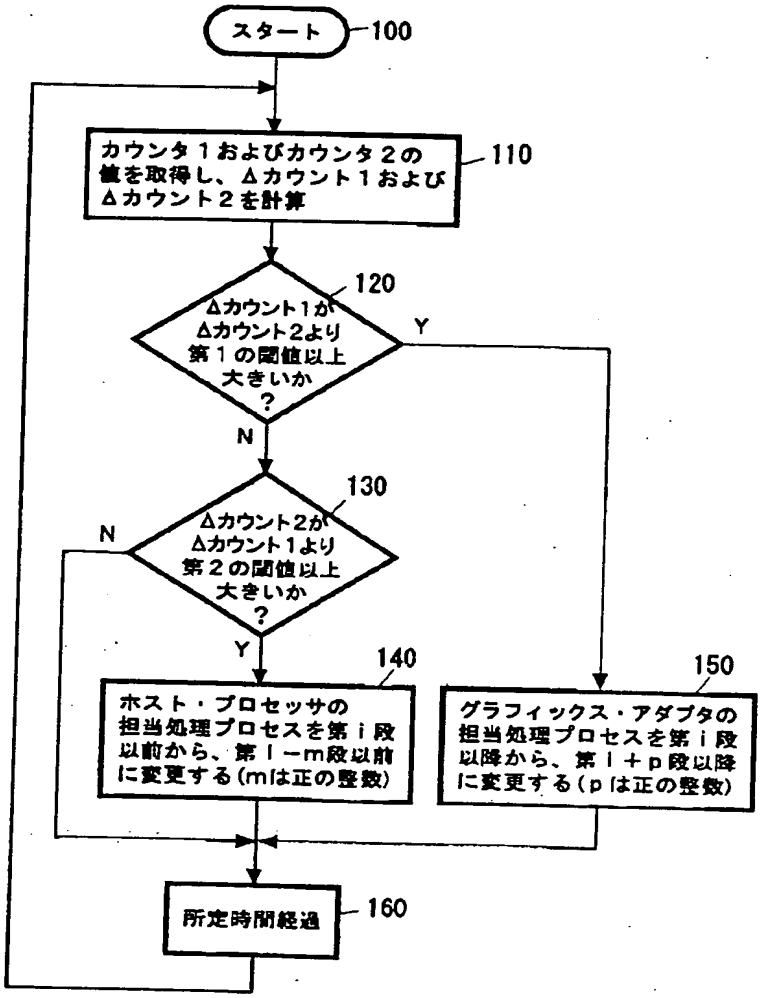
【図8】



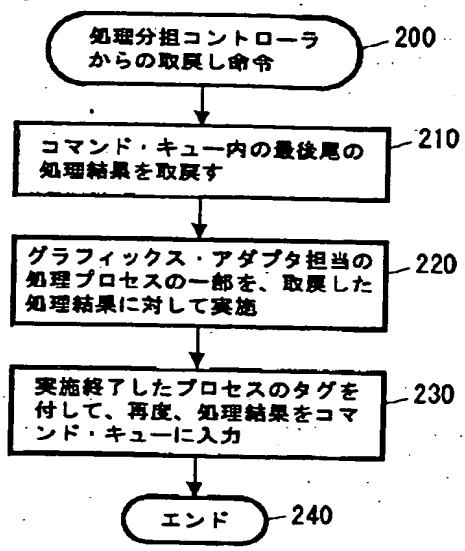
【図9】



【図6】


スタート
100
カウンタ1およびカウンタ2の値を取得し、Δカウント1およびΔカウント2を計算
110
Δカウント1がΔカウント2より第1の閾値以上大きいか？
120
Y
N
Δカウント2がΔカウント1より第2の閾値以上大きいか？
130
N
Y
ホスト・プロセッサの担当処理プロセスを第i段以前から、第i−m段以前に変更する(mは正の整数)
140
グラフィックス・アダプタの担当処理プロセスを第i段以降から、第i+p段以降に変更する(pは正の整数)
150
所定時間経過
160

【図7】



フロントページの続き

(72)発明者　川瀬　桂
神奈川県大和市下鶴間1623番地14　日本アイ・ビー・エム株式会社　東京基礎研究所内

(72)発明者　森山　孝男
神奈川県大和市下鶴間1623番地14　日本アイ・ビー・エム株式会社　東京基礎研究所内

(72)発明者　中村　英史
神奈川県大和市下鶴間1623番地14　日本アイ・ビー・エム株式会社　東京基礎研究所内

Fターム(参考)　5B045 AA01 GG02
5B057 AA01 CH02 CH05